

Программирование Python

Кафедра ИВТ и ПМ
ЗабГУ

2018

План

Синтаксис языка

Выражения

Операторы

- Оператор присваивания

- Оператор вызова функции

- Условный оператор

Синтаксис

Синтаксис языка программирования — набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом.

Пример синтаксического правила.

Для вызова функции с параметром x следует указать имя функции (её идентификатор), затем в круглых скобках выражение - параметр. Допустимо ставить пробелы между идентификаторами и скобками.

`sin (x)`

Синтаксические ошибки

В компилируемых языках синтаксические ошибки выявляются во время компиляции программы. Компилятор может обнаружить все синтаксические ошибки в программе.

В интерпретируемых языках - во время её выполнения. При обнаружении синтаксической ошибки программа аварийно завершает свою работу. Интерпретатор как правило может указать только на одну (первую) синтаксическую ошибку.

Если же какая-то часть программы не вопленилась, то она не была проверена на наличие синтаксических ошибок.

Синтаксические ошибки. Примеры

`sin x`

Синтаксические ошибки. Примеры

`sin x`

пропущены круглые скобки вокруг `x`

`20 / 7 +`

Синтаксические ошибки. Примеры

`sin x`

пропущены круглые скобки вокруг x

`20 / 7 +`

пропущен операнд после оператора сложения

`x + 2 = 7`

Синтаксические ошибки. Примеры

`sin x`

пропущены круглые скобки вокруг `x`

`20 / 7 +`

пропущен операнд после оператора сложения

`x + 2 = 7`

слева от оператора присваивания обязательно должна быть только переменная!

Синтаксические ошибки

Интерпретатор Python указывает место и тип синтаксической ошибки.

```
File "program.py", line 1
```

```
    20 / 7 +  
        ^
```

```
SyntaxError: invalid syntax
```

Синтаксические ошибки

Интерпретатор Python указывает место и тип синтаксической ошибки.

```
File "program.py", line 1
    20 / 7 +
        ^
```

SyntaxError: invalid syntax

Синтаксическая ошибка в файле *program.py* в первой строчке.

Особенности синтаксиса языка Python

- ▶ После инструкций точка с запятой не ставится
- ▶ Одна строка - одна инструкция
- ▶ Конец строки - конец инструкции
- ▶ Пустые строки интерпретатором игнорируются

Особенности синтаксиса языка Python

Пример последовательности инструкций (операторов):

```
a = 42
```

```
b = 13
```

```
c = a + b
```

```
print("a + b = ", c)
```

Отдельные логические части последовательностей операторов можно разделять пустыми строками.

Для улучшения читаемости знаки операций можно отделять пробелами.

Выражения

Выражение (expression) - комбинация значений, констант, переменных, операций и функций, которая может быть интерпретирована в соответствии с правилами конкретного языка.

Арифметическое выражение - выражение результат которого имеет числовой тип (целый или вещественный)

Выражения

Примеры выражений

2

3 + 1

`x` *# x - объявленная ранее переменная*

`22 + x / 2 - y**0.5`

`a and b` *# a и b переменные логического типа*

`sin(pi)`

`input()`

Логические выражения

Логическое выражение — выражение результатом вычисления которой является «истина» или «ложь»

True

False

2 == 2

2 * x == 5

"war" == "peace"

not a

b or a and c

x > 10

z <= 0.5

sin(pi) > 0

Логические выражения

Операции с логическим типом
and, or, not, \wedge (xor)

```
x = 5
```

```
y = 6
```

```
z = 5
```

```
x < y and x == z  # True
```

```
x != y  # True
```

```
x == 6 or x == 7  # True
```

```
x < y < z  # False
```

```
True ^ False  # True
```

Пустой оператор

pass - оператор который не выполняет никаких действий

Используется там, где синтаксис требует наличия оператора, но алгоритм не требует выполнения действий.

Также может использоваться как "заглушка" когда нужно представить общий алгоритм, а его детали реализовать позднее.

Пустой оператор

```
import random from random
x = random()
if x > 0.5:
    pass # todo: ...
else:
    pass # todo: ...
```

```
# функция которая ничего не делает
# но в неё будет добавлен код в будущем
def my_function():
    pass
```

Outline

Синтаксис языка

Выражения

Операторы

Оператор присваивания

Оператор вызова функции

Условный оператор

Оператор присваивания

Оператор присваивания (assignment statement)- оператор связывающий переменную некоторым значением.

Формат:

`переменная` = `выражение`

Во время выполнении *операции присваивания* сначала вычисляется результат выражения, а затем он связывается с переменной.

Оператор присваивания. Примеры

ПРАВИЛЬНО

`x = 20`

`y = 20/8 - 6 + x`

`z = x`

`x = x + 1`

`y = sin(0)`

`# ничего не изменить`

`x = x`

НЕПРАВИЛЬНО

`20 = 7`

`x + 1 = 8`

`x + y = 9`

`sin(x) = 0`

Особенности синтаксиса языка Python

- ▶ Отступы - это часть синтаксиса языка
- ▶ Операторных скобок нет
- ▶ Уровни вложенности определяются отступами.
- ▶ Для одного уровня вложенности рекомендуется использовать отступ из 4-х пробелов.

```
if a > b:  
    print("a > b")
```

```
while a > b:  
    a = a + 1
```

Перед вложенными операторами всегда ставится двоеточие.

Outline

Синтаксис языка

Выражения

Операторы

Оператор присваивания

Оператор вызова функции

Условный оператор

Оператор вызова функции ()

Функция — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.

Формат

`идентификатор` (`параметры`)

идентификатор - идентификатор функции;

параметры - выражения разделённые запятой.

Параметры могут отсутствовать.

Оператор вызова функции ()

Примеры

```
input()  # нет параметров
```

```
print("Python!")  # один параметр
```

```
log(9, 3)  # два параметра  
# = 2.0
```

Примеры функции

```
# функцию (как и переменную) нужно сперва объявить  
def my_function():  
    print("This is my very fist function!")  
    print("2 x 2 = 5")
```

```
# а потом использовать - вызвать  
my_function()
```

```
# на экран будет выведено  
>>> This is my very fist function!  
>>> 2 x 2 = 5
```

Оператор вызова функции ()

Если написать () послед обычной переменной или значения, Python будет думать, что вы хотите вызывать это как функцию.

```
x = 42
```

```
>>> TypeError: 'int' object is not callable  
# нельзя вызывать переменную типа int как функцию
```

```
3.14()
```

```
>>> TypeError: 'float' object is not callable
```

Оператор вызова функции ()

- ▶ Каждая функция имеет своё количество параметров (аргументов).
- ▶ Некоторые функции могут принимать разное количество аргументов
например: `print()`, `print("Hello, World!")` и `print("Hello, "World!")`
- ▶ Функция ожидает, что переданные ей параметры будут иметь конкретный тип
Например параметров функции `sin()` должно быть число, а не строка
- ▶ Некоторые функции не имеют жестких требований к типам передаваемых им параметров

Outline

Синтаксис языка

Выражения

Операторы

Оператор присваивания

Оператор вызова функции

Условный оператор

Условный оператор

Сокращённая форма

```
if лог.выражение :  
    оператор1
```

Условный оператор

Полная форма

```
if лог.выражение :  
    оператор1  
else:  
    оператор2
```


Условный оператор

Операторов может быть несколько.
Все они должны выделяться отступами

```
if лог.выражение:
```

```
    оператор1
```

```
    оператор2
```

```
    оператор3
```

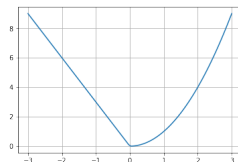
```
оператор4
```

оператор4 будет выполнен в любом случае.

Условный оператор. Примеры

Вычислить математическое выражение.

$$y = \begin{cases} x^2, & x > 0 \\ -2x, & x \leq 0 \end{cases}$$



```
x = float( input("x = ") )  
y = None
```

```
if x>0:  
    y = x**2  
else:  
    y = -3*x
```

```
print("y = ", y)
```

Условный оператор. Примеры

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

Условный оператор. Примеры

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b
```

```
print( "max = ", max )
```

Короткий вариант

```
max = a
if b > max:
    max = b
```

```
print( "max = ", max )
```

Условный оператор. Примеры

Логические выражения в условном операторе. В зависимости от ответа пользователя вывести заполнить переменную случайным число или с клавиатуры.

```
from random import random

ans = input("Заполнить данные с помощью random? ")
x = None
if ans == "да" or ans == 'lf':    # lf - да в англ. раскладке
    x = random()*100 - 50
    print(x)
else:
    x = float( input("Введите вещественное число: ") )
```

Условный оператор. Примеры

Разделение на два условных оператора

Если выражение внутри условного оператора это логическое И

Один условный оператор

```
...  
if x > 0 and y > 0:  
    print("Числа положительны")
```

Два условных оператора, один
вложен в другой

```
...  
if x > 0:  
    if y > 0:  
        print("Числа положительны")
```

Условный оператор. Примеры

В какой четверти координатной плоскости находится точка с заданными координатами (x, y) ?

Предположить что, координаты не могут быть нулями.

Условный оператор. Примеры

В какой четверти координатной плоскости находится точка с заданными координатами (x, y) ?

Предположить что, координаты не могут быть нулями.

Сколько понадобится условных операторов?

Условный оператор. Примеры

В какой четверти координатной плоскости находится точка с заданными координатами (x, y) ?

Предположить что, координаты не могут быть нулями.

Сколько понадобится условных операторов?

```
if x > 0:
    if y > 0:
        print("Первая четверть")
    else: # y < 0
        print("Четвёртая четверть")
else: # x < 0
    if y > 0:
        print("Вторая четверть")
    else: # y < 0
        print("Третья четверть")
```

Условный оператор. Примеры

В какой четверти координатной плоскости находится точка с заданными координатами (x, y) ?

Предположить что, координаты не могут быть нулями.

Условный оператор. Примеры

В какой четверти координатной плоскости находится точка с заданными координатами (x,y)?

Предположить что, координаты не могут быть нулями.

Вариант решения задачи со сложным условием.

```
if x > 0 and y > 0:
    print("Первая четверть")
else:
    if x > 0 and y < 0:
        print("Четвёртая четверть")
    else:
        if x < 0 and y > 0:
            print("Вторая четверть")
        else: # хм, остался только один вариант
               # значит эта ветка для x<0 and y<0
            print("Третья четверть")
```

Такой вариант кода немного труднее для чтения и понимания.

Условный оператор. Примеры

Определить находится ли точка в четвёртой четверти координатной плоскости.

Вариант 1

```
if x > 0 and y < 0:  
    print("Да, четвёртая четверть")  
else:  
    print("Нет")
```

Вариант 2

```
if x > 0:  
    if y < 0:  
        print("Да, четвёртая четверть")  
    else:  
        print("Нет")  
else:  
    print("Нет")
```

Условный оператор. Примеры

Определить находится ли точка в четвёртой четверти координатной плоскости.

Вариант 1

```
if x > 0 and y < 0:  
    print("Да, четвёртая четверть")  
else:  
    print("Нет")
```

Вариант 2

```
if x > 0:  
    if y < 0:  
        print("Да, четвёртая четверть")  
    else:  
        print("Нет")  
else:  
    print("Нет")
```

А здесь какой из вариантов лучше?

Оператор выбора

Что если нужно совершить то или иное действие в зависимости от значения одной переменной?

Например в зависимости от имени вывести на экран правильно обращение.

Александр -> Здравсуйте, уважаемый Александр!
Александра -> Здравсуйте, уважаемая Александра!
Саша -> Здравсуй, Саша!

Оператор выбора

```
if name == "Александр":  
    print("Здравсуйте, уважаемый", name, "!")  
else:  
    if name == "Александра":  
        print("Здравсуйте, уважаемая", name, "!")  
    else:  
        if name == "Саша":  
            print("Здравствуй, ", name, "!")
```

Оператор выбора

Чтобы сократить код можно было бы использовать оператор выбора.

В котором каждому из вариантов поставить в соответствие подходящее обращение в `print`

Оператор выбора

Чтобы сократить код можно было бы использовать оператор выбора.

В котором каждому из вариантов поставить в соответствие подходящее обращение в `print`

Но, оператора выбора в Python нет :(

Оператор выбора

Чтобы сократить код можно было бы использовать оператор выбора.

В котором каждому из вариантов поставить в соответствие подходящее обращение в print

Но, оператора выбора в Python нет :(

Зато есть оператор **elif** который можно использовать вместо `else + if`

Оператор elif

```
if name == "Александр":  
    print("Здравсуйте, уважаемый", name, "!")  
elif name == "Александра":  
    print("Здравсуйте, уважаемая", name, "!")  
elif name == "Саша":  
    print("Здравствуй, ", name, "!")
```

Все операторы на одном уровне вложенности и код читается легче. Особенно если расположить выражения в условии строго одно под другим.

Оператор elif

elif как и оператор if можно дополнить оператором else.

```
if name == "Александр":  
    print("Здравсуйте, уважаемый", name, "!")  
elif name == "Александра":  
    print("Здравсуйте, уважаемая", name, "!")  
elif name == "Саша":  
    print("Здравствуй, ", name, "!")  
else:  
    # оператор else работает как обычно  
    print("Здравствуй!")
```

Рекомендации

- ▶ Создавайте короткие и понятные условные операторы
- ▶ Проверяйте, все ли варианты учтены
- ▶ Используйте алгебру логики (алгебру высказываний) для упрощения логических выражений

Ссылки и литература

- ▶ ru.wikibooks.org/wiki/Python - Викиучебник
- ▶ Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- ▶ Официальная документация Python3
`help(имя)`

Ссылки и литература

Материалы курса
github.com/VetrovSV/Programming